CSci 6974 and ECSE 6966 Math. Tech. for Vision, Graphics and Robotics
Lecture 21, April 17, 2006
Estimating A Plane Homography

Overview

We continue with a discussion of the major issues, using estimation of plane projective transformations (homographies) as an example. We will discuss:

- The basic problem and the DLT algorithm.
- Distance metrics
- Objective functions
- Normalization

Homography (Plane Projective Transform) Estimation

- Consider two perspective images of a planar surface.
- Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be the image projections of a set of points on this plane in one image, and let $\{\mathbf{x}'_1, \ldots, \mathbf{x}'_n\}$ be the projections of the same set of points in the second image.
  - We will assume we know that the points $\mathbf{x}_i$ and $\mathbf{x}'_i$ correspond to each other without saying how we know this (and avoiding a hard problem).
- The problem is to estimate the parameters of the matrix $\mathbf{H}$ that maps $\mathbf{x}_i$ as close to $\mathbf{x}'_i$ as possible.
- Recall that

$$\mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \tag{1}$$

  and is defined up to non-zero scale factor. Hence $\mathbf{H}$ has 8 degrees of freedom.

Direct Linear Transform (DLT) Algorithm

For the following discussion, write

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}_1^\top \\ \mathbf{h}_2^\top \\ \mathbf{h}_3^\top \end{pmatrix}. \tag{2}$$

- $\mathbf{H}$ maps $\mathbf{x}_i$ onto $\mathbf{x}'_i$ if
$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \mathbf{0} \tag{3}$$
- Writing $\mathbf{x}'_i = (u'_i, v'_i, w'_i)^\top$, this constraint is expressed as
$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \begin{pmatrix} v'_i \mathbf{h}_3^\top \mathbf{x}_i - w'_i \mathbf{h}_2^\top \mathbf{x}_i \\ w'_i \mathbf{h}_1^\top \mathbf{x}_i - u'_i \mathbf{h}_3^\top \mathbf{x}_i \\ u'_i \mathbf{h}_2^\top \mathbf{x}_i - v'_i \mathbf{h}_1^\top \mathbf{x}_i \end{pmatrix} = \mathbf{0}, \tag{4}$$
with obvious simplifications when the points are normalized so that $w'_i = 1$.
- Rearranging, this can be written as
$$\begin{pmatrix} \mathbf{0}^\top & -w'_i \mathbf{x}_i^\top & v'_i \mathbf{x}_i^\top \\ w_i \mathbf{x}_i^\top & \mathbf{0}^\top & -u'_i \mathbf{x}_i^\top \\ -v'_i \mathbf{x}_i^\top & u'_i \mathbf{x}_i^\top & \mathbf{0}^\top \end{pmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = \mathbf{A}_i \mathbf{h} = \mathbf{0}. \tag{5}$$
Notice we have stacked the parameters of $\mathbf{H}$ into the 9 component column vector $\mathbf{h}$.
- $\mathbf{A}_i$ is 3×9 and expresses the constraints on $\mathbf{h}$ from correspondence i. It is only rank 2, however, because the 3^{rd} row can be written as a linear combination of the first two. We could therefore eliminate the last row.
- We need 4 point correspondences to establish the minimum of 8 constraints on $\mathbf{h}$ (remember, scale does not matter). This gives
$$\begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{pmatrix} \mathbf{h} = \mathbf{A}\mathbf{h} = \mathbf{0}. \tag{6}$$
- The solution is the unit vector spanning the nullspace of $\mathbf{A}$ (assuming it is rank 1). Of course this can be found from the SVD of $\mathbf{A}$.
- Existence and uniqueness conditions:
  - When three points are collinear in I, a rank 1 matrix $\mathbf{H}^*$ satisfies the conditions $\mathbf{x}'_i \times \mathbf{H}^* \mathbf{x}_i = \mathbf{0}$, $i \in \{1, 2, 3, 4\}$.
  - This matrix $\mathbf{H}^*$ does not satisfy $\mathbf{x}'_i = \mathbf{H}^* \mathbf{x}_i$.

  - In general, if no three of the points are collinear, there is a unique solution.

- When more than four point correspondences are involved, then the constraints from each point are stacked in $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \cdots \\ \mathbf{A}_k \end{pmatrix} \tag{7}$$

  and the least-squares DLT solution is found using the SVD as earlier.

- See Section 3.1 (4.1 in the new edition) of Hartley and Zisserman for more details and discussion of different constraints.

- It is important to note that the practical results of applying this algorithm are **VERY POOR**. The rest of these lecture notes lead to a better estimation technique.

Distance Metrics: Algebraic and Geometric Distance

Our next consideration is the point distance measures

- Stating things a bit abstractly, we are given some kind of data point vector $\mathbf{p}$, a parameter vector $\mathbf{a}$, and an implicit function $f(\mathbf{p}; \mathbf{a})$ representing an "algebraic variety" such that

$$f(\mathbf{p}; \mathbf{a}) = 0 \tag{8}$$

  if and only if point $\mathbf{p}$ is on the variety determined by parameter vector $\mathbf{a}$ and function f.

- The *algebraic distance* of a point $\mathbf{p}$ is just

$$\mid f(\mathbf{p}; \mathbf{a}) \mid \tag{9}$$

  This distance is zero for points on the variety and increases for points not on the variety.

- Examples:
  - For lines in two dimensions, $\mathbf{p} = (x, y, 1)^\top$, $\mathbf{a} = (a_0, a_1, a_2)^\top$ and

$$f(\mathbf{p}; \mathbf{a}) = \mathbf{p}^\top \mathbf{a}. \tag{10}$$

  - For conics in two dimensions, $\mathbf{p} = (x, y, 1)^\top$, $\mathbf{a}$ is the parameters of the conic matrix $\mathbf{C}$ and

$$f(\mathbf{p}; \mathbf{a}) = \mathbf{p}^\top \mathbf{C} \mathbf{p}. \tag{11}$$

– For the plane homography, $\mathbf{p} = (u_i, v_i, u_i', v_i')^\top$, $\mathbf{a} = (\mathbf{h}_1^\top, \mathbf{h}_2^\top, \mathbf{h}_3^\top)^\top$, and

$$f(\mathbf{p}; \mathbf{a} = (\mathbf{h}_1^\top \mathbf{x} - v'\mathbf{h}_3^\top)^2 + (\mathbf{h}_2^\top \mathbf{x} - v'\mathbf{h}_3^\top)^2 \tag{12}$$

This is also $\|\mathbf{A}_i\mathbf{h}\|^2$ from above (when we only consider the first two rows of $\mathbf{A}_i$).

- The *geometric distance* is simply the distance from $\mathbf{p}$ to the nearest point $\mathbf{p}^*$ such that $f(\mathbf{p}^*; \mathbf{a}) = 0$.
- This sets up a problem in Lagrange multipliers, just like we solved for lines, circles and conics. In other words, the distance between a point $\mathbf{x}$ and the variety is found by finding the $\mathbf{p}$

$$F(\mathbf{p}, \lambda) = \|\mathbf{p} - \mathbf{x}\|^2 - 2\lambda f(\mathbf{p}; \mathbf{a}), \tag{13}$$

and then the distance is simply

$$\|\mathbf{p} - \mathbf{x}\|. \tag{14}$$

- In some cases, the relationship between the algebraic and geometric distances is straightforward. Examples include the distances of points to lines and planes. Even stronger, for the affine transform estimation the algebraic and geometric errors are the same!
- Good approximations to the geometric distance can be obtained using the what's known as the "Sampson error". We will discuss this next.

Sampson Error

- Write $\Delta\mathbf{x} = \mathbf{p} - \mathbf{x}$. Then the Lagrangian form in (13) may be rewritten as

$$F(\Delta\mathbf{x}, \lambda) = \|\Delta\mathbf{x}\|^2 - 2\lambda f(\mathbf{x} + \Delta\mathbf{x}; \mathbf{a}), \tag{15}$$

- Taking the derivative with respect to $\Delta\mathbf{x}$ and λ yields the simultaneous equations

$$\Delta\mathbf{x} - \lambda\nabla f(\mathbf{x} + \Delta\mathbf{x}; \mathbf{a})^\top = \mathbf{0} \tag{16}$$

$$f(\mathbf{x} + \Delta\mathbf{x}; \mathbf{a}) = 0 \tag{17}$$

(Remember, gradient vectors, ∇f, are *row vectors*; hence the need for the transpose in the first equation.)

- To obtain the Sampson error, two approximations are required:
  - In the first term, the gradient vector is evaluated at $\mathbf{x}$ instead of $\mathbf{x} + \Delta\mathbf{x}$. This gradient will be denoted simply as ∇f.
  - The second term is replaced by a first order Taylor expansion.

- Combined, these results yield the simplified equations:

$$\Delta \mathbf{x} - \lambda \nabla f^\top = \mathbf{0} \tag{18}$$

$$f(\mathbf{x}; \mathbf{a}) + \nabla f \Delta \mathbf{x} = 0 \tag{19}$$

- These two equations may be solved to yield the Sampson distance approximation:

$$\|\Delta x\| = \frac{f(\mathbf{x}; \mathbf{a})}{(\nabla f \nabla f^\top)^{1/2}} \tag{20}$$

- This is the algebraic distance normalized by the gradient magnitude.
  - Notice that this is the exact Euclidean distance for lines in $\mathbb{R}^2$ and planes in $\mathbb{R}^3$.

Possible Objective Functions for Homography Estimation

Building from our understanding of distances, we can think of a variety of objective functions that can be minimized in plane homography estimation:

- Algebraic distance:

$$\sum_i \|\mathbf{A}_i \mathbf{h}\|^2 \tag{21}$$

  where $\mathbf{A}_i$ is constructed from $\mathbf{x}_i$ and $\mathbf{x}'_i$ as above.

- Sampson error, to be derived in class.

- Geometric distance in one image

$$\sum_i d(\mathbf{x}'_i, \mathbf{H}\mathbf{x}_i)^2 \tag{22}$$

  where $d(\mathbf{x}'_i, \mathbf{H}\mathbf{x})$ is the geometric distance.

- Symmetric transfer error:

$$\sum_i d(\mathbf{x}'_i, \mathbf{H}\mathbf{x}_i)^2 + d(\mathbf{x}'_i, \mathbf{H}^{-1}\mathbf{x}_i)^2 \tag{23}$$

- Reprojection error. Here, the "true" point locations are estimated as "nuisance parameters" to estimating the parameters of $\mathbf{H}$:

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \qquad \text{subject to} \qquad \hat{\mathbf{x}}'_i = \mathbf{H}\hat{\mathbf{x}}_i, \forall i \tag{24}$$

- Most of these lead to non-linear minimization problems. These methods will be discussed in a later lecture. For the rest of this lecture we will consider an important way to improve the simple DLT / algebraic error technique.

Normalization, Normalization, NORMALIZATION!

- As a motivating example, consider the simple case of fitting a parabola $y = ax^2 + bx + c$ to a set of points $\{(x_i, y_i)\}$. This leads to the least-squares minimization problem

$$\|\mathbf{X}\mathbf{a} - \mathbf{y}\|^2, \tag{25}$$

where $\mathbf{a} = (a, b, c)^\top$, $\mathbf{y} = (y_1, \ldots, y_n)^\top$, and

$$\mathbf{X} = \begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{pmatrix} \tag{26}$$

The important thing to note here is the different units in the different columns of $\mathbf{X}$. Think of the effect of a small error in $\mathbf{x}_i$ on the entries of each of the 3 different columns.

- A similar situation arises in the terms of the matrix $\mathbf{A}$ in the DLT homography estimation:

$$\mathbf{A}_i = \begin{pmatrix} \mathbf{0}^\top & -\mathbf{x}_i^\top & v_i'\mathbf{x}_i^\top \\ \mathbf{x}_i^\top & \mathbf{0}^\top & -u_i'\mathbf{x}_i^\top \\ -v_i'\mathbf{x}_i^\top & u_i'\mathbf{x}_i^\top & \mathbf{0}^\top \end{pmatrix} \tag{27}$$

- These differences become problems — both statistically and numerically — when the data are corrupted by noise.

- The most straightforward approach is normalization, which should ALWAYS be used, even with other methods. More sophisticated techniques directly address the "heteroscedasticity" (non-stationary noise) in the data.

- The standard technique is to center and normalize the values in each column separately.

- For example, in parabola fitting:

  1. Compute $\mu_2 = \sum_i x_i^2/N$, $c_2 = \sum_i |x_i^2 - \mu_2|/N$, $\mu_1 = \sum_i x_i/N$, and $c_1 = \sum_i |x_i - \mu_1|/N$.
  2. Replace $\mathbf{X}$ by

$$\mathbf{X}' = \begin{pmatrix} (x_1^2 - \mu_2)/c_2 & (x_1 - \mu_1)/c_1 & 1 \\ (x_2^2 - \mu_2)/c_2 & (x_2 - \mu_1)/c_1 & 1 \\ \vdots & \vdots & \vdots \\ (x_n^2 - \mu_2)/c_2 & (x_n - \mu_1)/c_1 & 1 \end{pmatrix} \tag{28}$$

  3. Solve the least-squares minimization problem

$$\|\mathbf{X}'\mathbf{a}' - \mathbf{y}\|^2 \tag{29}$$

  4. Recover $\mathbf{a}$ from $\mathbf{a}'$ and μ_2, c_2, μ_1, c_1.

Normalized DLT: A Much Better Solution

Hartley and Zisserman recommend a slightly different approach for normalization in the DLT estimation of $\mathbf{H}$:

1. Normalize each u_i, v_i, u'_i and v'_i separately so that the average magnitude of each is 1.

   - If μ_u, μ_v, $\mu_{u'}$ and $\mu_{v'}$ are the centers, and s_u, s_v, $s_{u'}$ and $s_{v'}$ are the scaling factors, then the normalization process may be expressed as the affine transformations

   $$\mathbf{x}_i^* = \mathbf{T}\mathbf{x}_i \qquad \text{and} \mathbf{x}_i'^* = \mathbf{T}'\mathbf{x}'_i \tag{30}$$

   where

   $$\mathbf{T} = \begin{pmatrix} 1/s_u & 0 & -\mu_u/s_u \\ 0 & 1/s_v & -\mu_v/s_v \\ 0 & 0 & 1 \end{pmatrix} \tag{31}$$

   and

   $$\mathbf{T}' = \begin{pmatrix} 1/s_{u'} & 0 & -\mu_{u'}/s_{u'} \\ 0 & 1/s_{v'} & -\mu_{v'}/s_{v'} \\ 0 & 0 & 1 \end{pmatrix} \tag{32}$$

   - Apply the DLT algorithm by forming matrix $\mathbf{A}$ from the $\mathbf{x}_i^*$ values $\mathbf{x}_i'^*$. Call the resulting matrix $\mathbf{H}^*$.
   - The desired DLT solution to the original problem is

   $$\mathbf{H} = \mathbf{T}'^{-1}\mathbf{H}^*\mathbf{T}. \tag{33}$$

2. We will have a homework assignment involving the implementation of the DLT and the normalized DLT algorithm. My experience is that this a problem and solution you have to see to believe!

Practice/Test Problems

1. Show that the algebraic distance and geometric distance are the same for estimating an affine transformation.

2. Derive the Sampson error approximation for the distance of a point $\mathbf{x}$ to a conic.

3. Derive the normalized equations for a least-squares method to estimate the parameters of a conic. Use each of the two methods described above.

Homework Problems

People have trouble seeing the significance of normalization until they try it practice. Therefore, in this problem you will write a program that implements the DLT algorithm for homogeneous transformation estimation. You will implement it with and without normalization. I strongly urge you to use Matlab or Maple or a good C++ library that has an SVD implementation.

The data will be provided as a text file. Each line of input of this file will have four floating point values, x_i, y_i, x'_i, y'_i, corresponding to a pair of corresponding image locations. Data will be available on the course web page.

The output from your program should be the parameters of the estimated matrix $\mathbf{H}$ for the DLT and the normalized DLT algorithms, and two plots. Let $\mathbf{H}_D$ and $\mathbf{H}_N$ be the two estimated matrices. The first plot should contain the points (x'_i, y'_i) and the points $\mathbf{H}_D(x_i, y_i, 1)^T$. The second plot should contain the points (x'_i, y'_i) and the points $\mathbf{H}_N(x_i, y_i, 1)^T$. The $matxH$ from which the points were generated will be posted on the web.

Submit printed copies of your source code and your output. Your code should be clean and well-documented.